Week 11 : Visualisation

pending tasks	
type	

Data visualisation

- Data visualisation is about describing data.Compressing the information to be focused on from the available data can be done by computing statistics from it or visualizing the data. Data visualization is visual encoding of data that can be perceived by humans. This is done to discover insights from the data and communicate the insights effectively
- Visual elements like length, slopes, colours, volumn, angles, lengths(align), area and colour intensity can be used.The goal is to reduce perceptual errors.
- The visual aids used have to be concise and easy to understand. It should not have multiple dimensions or too much information encoded into a single charts, making it hard to interpret.
- Effective visualisation involves syntax of the libraries, understanding plot types and usage, aesthetics of plot and communicating with plots.
- The packages used are seaborn and matplotlib along with some ideas in pandas.

Focus on understanding plot types from first principles and then seeing examples of their usage.

Read complex JSON files

- The json file can be downloaded using request from urllib library as follows:

```
url = 'https://api.covid19india.org/states_daily.json'
urllib.request.urlretrieve(url, 'data.json');
```

- The file is preprocessed using dictionary, before loading it into a dataframe using json_normalize().

```
with open('data.json') as f:
    data = json.load(f)
data = data['states_daily']
covid_data = pd.json_normalize(data)
```

- Convert the date column to datetime type and then into index, filter the required data and drop the unnecessary columns.

```
df.date = pd.to_datetime(df.date)
df = df[df.status == 'Confirmed']
df.drop('status', axis=1, inplace=True)
df.set_index('date', inplace=True)
```

- Numeric type is more efficient for data processing than object type. Thus, the required object type columns with numbers should be converted to numeric types. To apply a function of conversion to numeric type to all columns, .apply() method can b used as follows:

```
df = df.apply(pd.to_numeric)
```

Styling tabulation

1. Highlighting all negative values:

```
def colour_red_negative(x):
    color = 'red' if x < 0 else 'white'
    return 'color: ' + color
df.style.applymap(colour_red_negative)
```

Useful to have an overview of values in each column at a glance, dropping the ones with most negative values.

2. Highlighting minimum and maximum in each column

```
df.style.highlight_max(color='red')
df.style.highlight_max(color='red').highlight_min(color='green')
```

3. Bold the maximum values

```
def bold_max_value(x):
    is_max = (x == x.max())
    return ['font-weight: bold' if y else '' for y in is_max]
df.style.apply(bold_max_value)
# stringing together with highlight fn
df.style.apply(bold_max_value).highlight_min(color='green')
```

NOTE : **apply** works on a row / column basis of a DataFrame, **applymap** works element-wise on a DataFrame, and **map** works element-wise on a Series.

4. Operations can be done either column-wise or row-wise. In the example below, row-wise max is highlighted in red while column-wise max is bold.

```
df.style.apply(bold_max_value).highlight_max(color='red', axis=1)
```

5. Gradient colours across cell values can be set along an axis

```
# column-wise
df.style.background_gradient(cmap='Reds')
# row-wise
df.style.background_gradient(cmap='Reds', axis=1)
```

6. Selecting and highlighting subsets of data.

```
df.style.background_gradient(cmap='Reds', subset=['mh', 'tn', 'dl'])
```

7. Drawing bars, happens column-wise and subsets can be selected.

```
# column-wise bars
df.style.bar()
# subset of columns approach-1
df.style.bar(subset=['mh', 'tn', 'dl'])
# subset of columns approach-2
df[['mh', 'tn', 'dl']].style.bar()
# assigning different shades to the subsets
df[['mh', 'tn', 'dl']].style.bar(subset=['mh'], color='red').bar(subset=['tn'], color='orange').bar(subset=['dl'], color='yellow')
```

Distribution of data - histograms

1. Example syntax (as discussed in the lecture)

distplot:

kde : bool, optional

Whether to plot a gaussian kernel density estimate.

rug : bool, optional

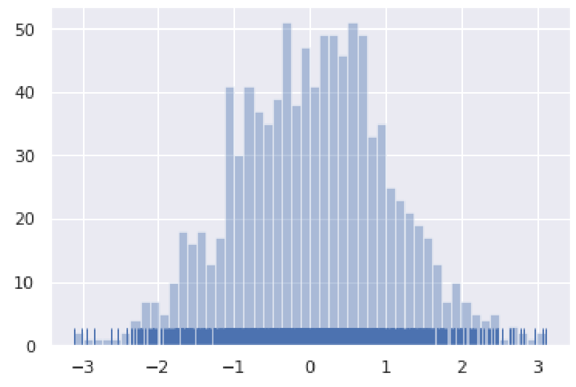
Whether to draw a rugplot on the support axis.

bins : argument for matplotlib hist(), or None, optional

Specification of hist bins. If unspecified, as reference rule is used that tries to find a useful default.

```
x = np.random.normal(size=1000)
y = np.random.uniform(size=1000)
```

```
sns.set(color_codes=True)
sns.distplot(x, kde=False, rug=True, bins=50);
```

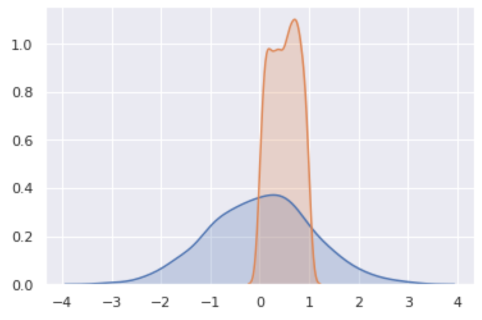


kde plot

shade : bool

Alias for fill. Using fill is recommended.

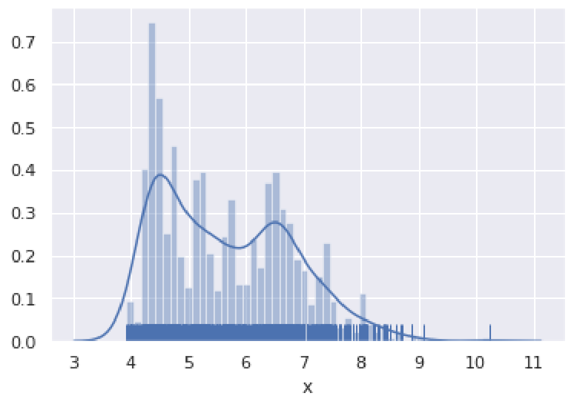
```
# kde plots
sns.kdeplot(x,shade=True)
sns.kdeplot(y,shade=True);
```



2. Example of real world data - diamond dataset

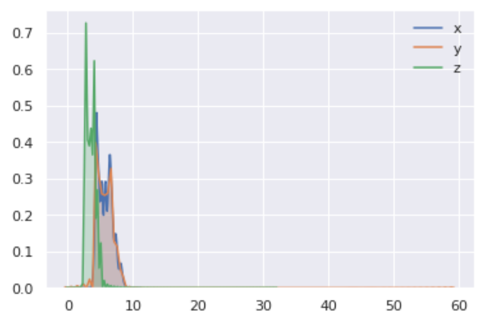
- Plotting carat is an example of multimodal distribution.
- Plotting price is a right skewed long tailed distribution.
- rug helps see the data points plotted.

```
d = sns.load_dataset('diamonds')
sns.distplot(d.sample(1000).x, rug=True, bins=50);
```



- kde plots are used to plot multiple columns at once for comparison. Shading the region helps give a better sense of the distribution.

```
# kde plot
sns.kdeplot(d.x, shade=True)
sns.kdeplot(d.y, shade=True)
sns.kdeplot(d.z, shade=True);
```



There are three types of data distribution discussed :

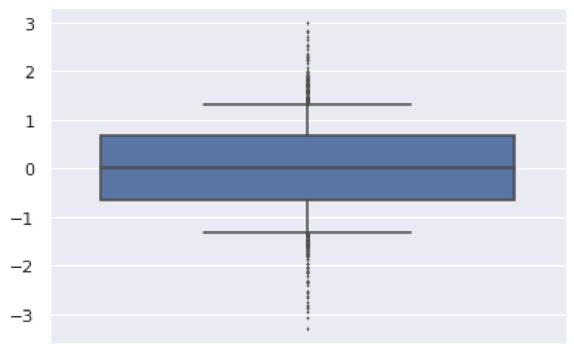
1. Distribution of single continuous variable - histograms, box plot, boxen plots.
2. Distribution of categorical variable - bar plots
3. Joint distribution of two variables - jointplot, swarm plot, violin plot, faceted plotting, pair plot.

Box plot

- Box plots are used to visualize the distribution to 50% of data (between Q1 and Q3) , the median and the outliers.
- Setting the whisker parameter helps set the tolerance for outliers. Seaborn takes 1.5 times the IQR as whisker length and the points beyond the whiskers are the outliers.
- Plotting multiple boxplots is done vertically.

1. Example syntax

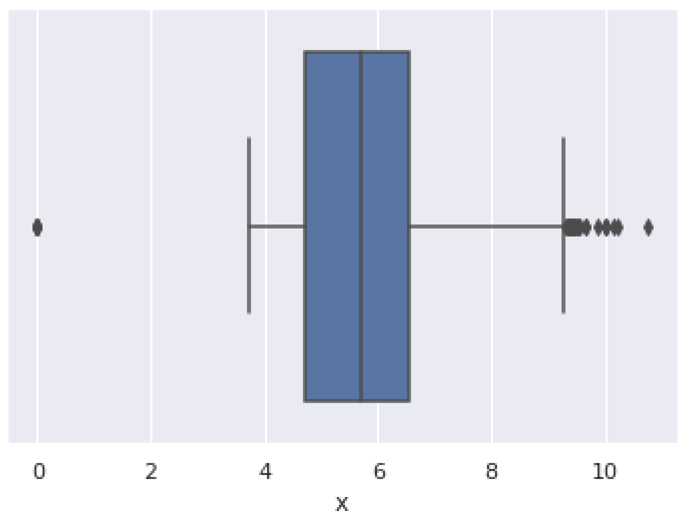
```
x = np.random.normal(size=1000)
sns.boxplot(x, whis=0.5, fliersize=1, orient='v');
```

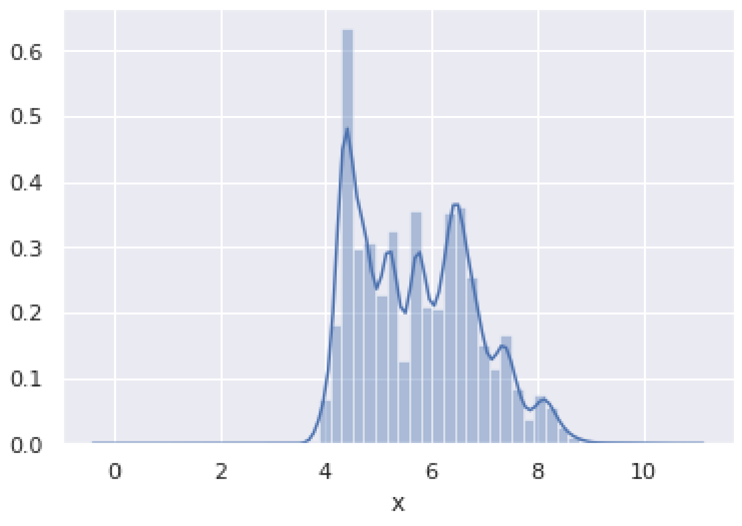


2. Associating boxplot with distplots for diamonds dataset

- The multimodal distribution of x is not visible in box plot but can be seen in the distribution plot.

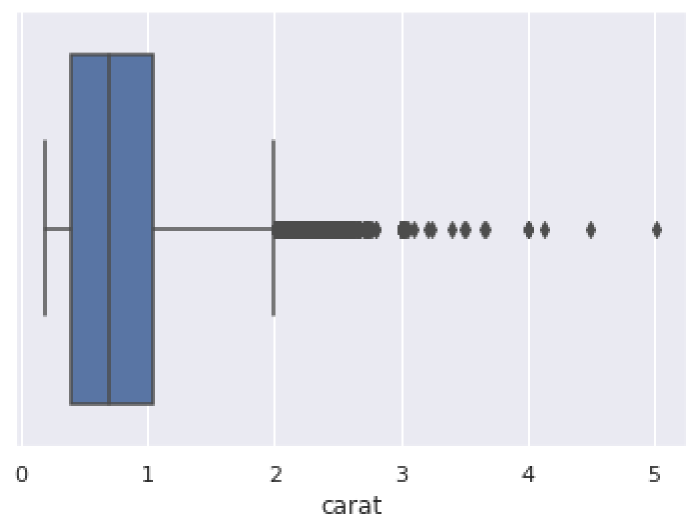
```
sns.boxplot(d.x);
sns.distplot(d.x);
```

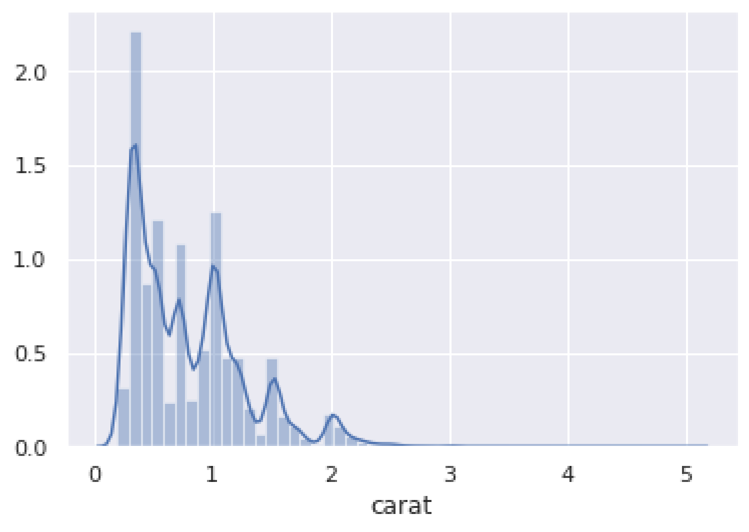




- The skewness of the carat can be realised from the boxplots. The distribution has a long tail and thus, many outliers after the right whisker. The width of the box can be used to understand the distribution around the median(e.g. uniform distribution has a broad box whereas normal distribution has a thin box).

```
sns.boxplot(d.carat)
sns.distplot(d.carat)
```



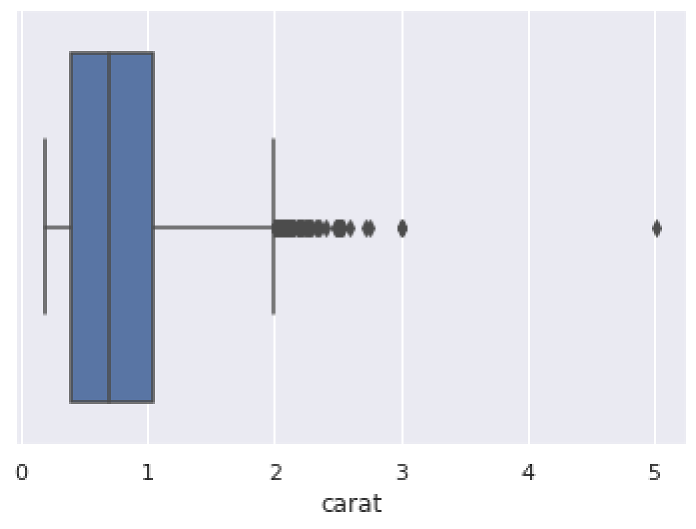


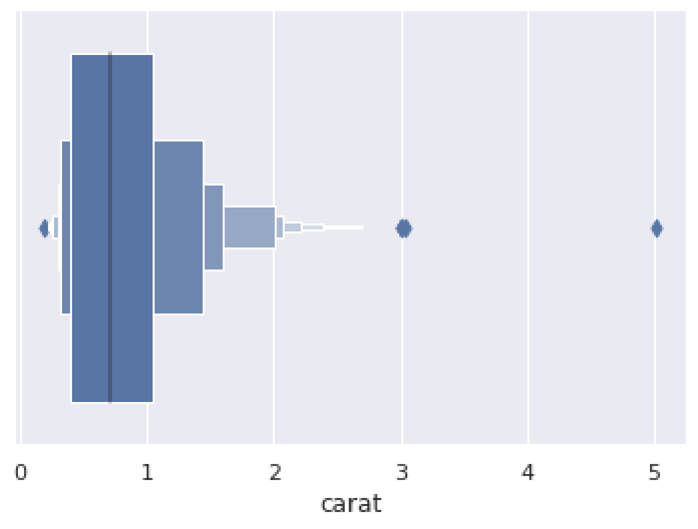
Boxen plots

1. Boxen plots vs box plots - using diamonds dataset

- In box plots the distribution of data between the whiskers and the respective quartiles cannot be known. The boxen plots give a better representation of the distribution by displaying more quantiles of data.
- By default the quantiles to be plotted are calculated by progressively halving the previous quantile e.g - 0.50,0.25,0.125 and so on.
- If the data does not have too many boxes and is spaced out, then , using boxen plots is better than using box plots.

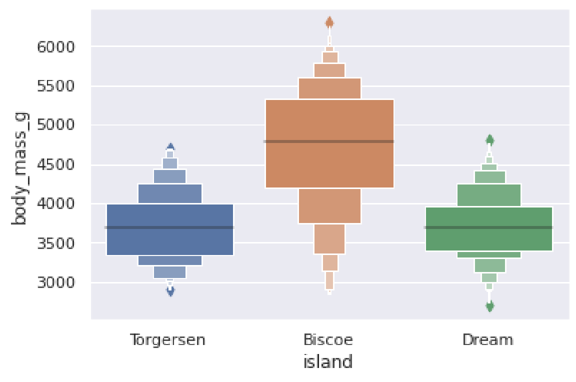
```
# first plot - boxplot
sns.boxplot(d.sample(5000).carat);
# second plot - c=boxen plot
sns.boxenplot(d.sample(5000).carat);
```





2. Example distribution of a continuous variable over a categorical variable - using diamonds dataset.

```
sns.boxenplot(x = 'island', y = 'body_mass_g', data = p);
```

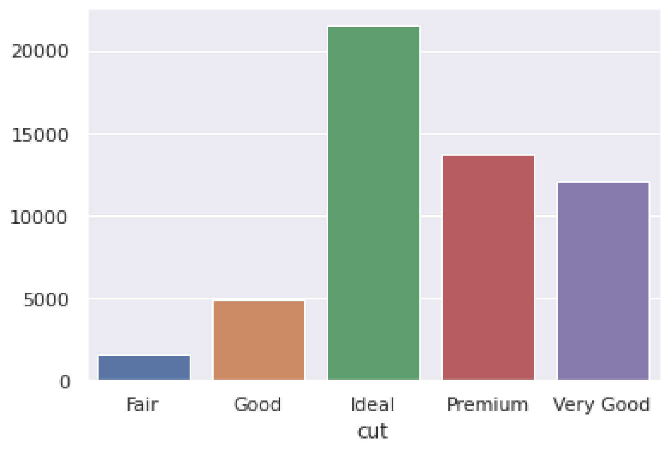


Distribution of a categorical variable (Box Plots)

- Distribution of discrete variables can be visualised using bar plots.
- The input to bar plots is a series with counts of each bar to be potted.

An example using the diamonds data set is given below:

```
c = d.groupby('cut')['cut'].count()
sns.barplot(x=c.index, y=c.values)
```



Joint distribution of two variables

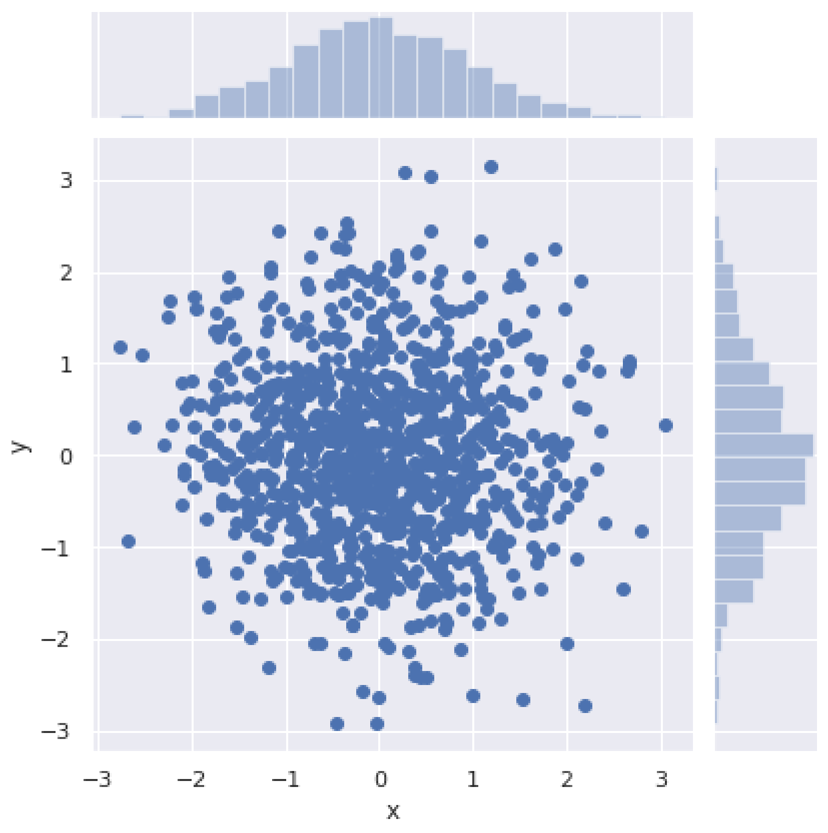
These are used to visualise the relationship between two variables.

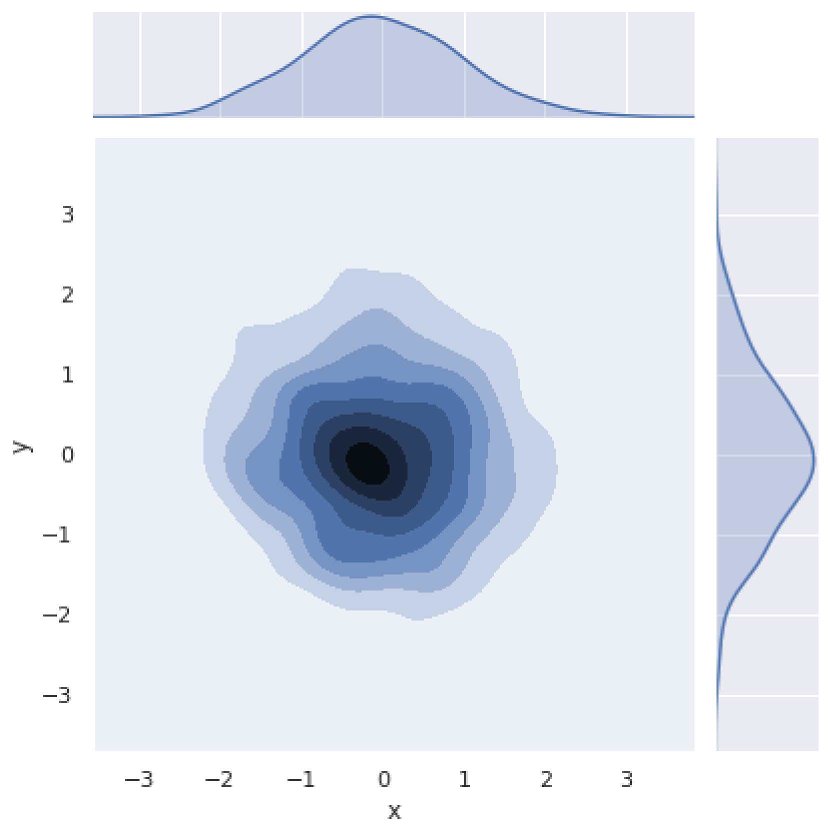
Jointplot

- Jointplots can be used to view the relationship between two variables along with their corresponding distributions. Scatter plot or kde plots are used for this.
- Lists of values can be passed as x and corresponding y values or columns of a dataframe can be passed.

1. Example syntax:

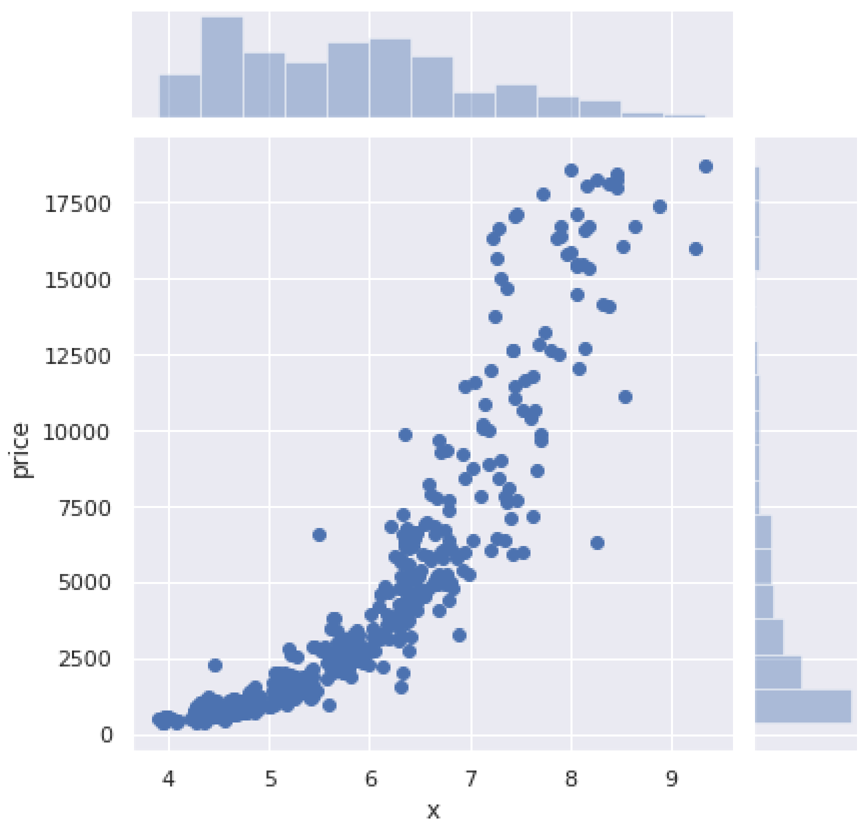
```
x = np.random.normal(size=1000)
y = np.random.normal(size=1000)
# first plot
sns.jointplot('x', 'y', data=df);
# second plot
sns.jointplot('x', 'y', data=df, kind='kde');
```

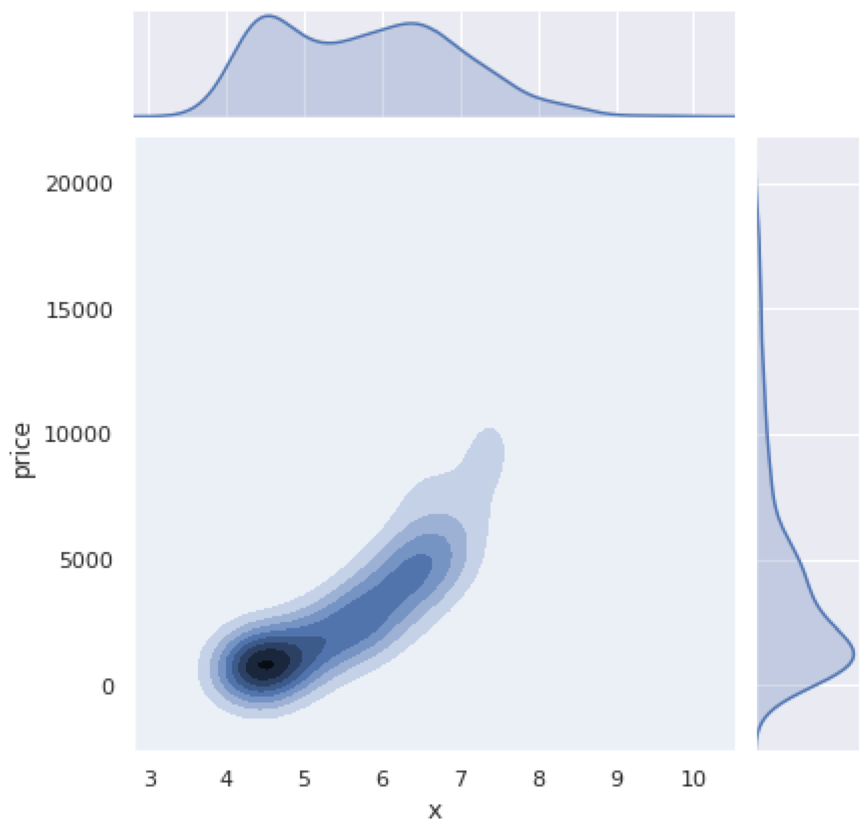




2. Example using diamonds dataset, price is plotted against x and a quadratic relationship can be observed between the two variables.

```
# first plot
sns.jointplot('x', 'price', data=d.sample(500));
#  second plot
sns.jointplot('x', 'price', data=d.sample(500), kind='kde');
```



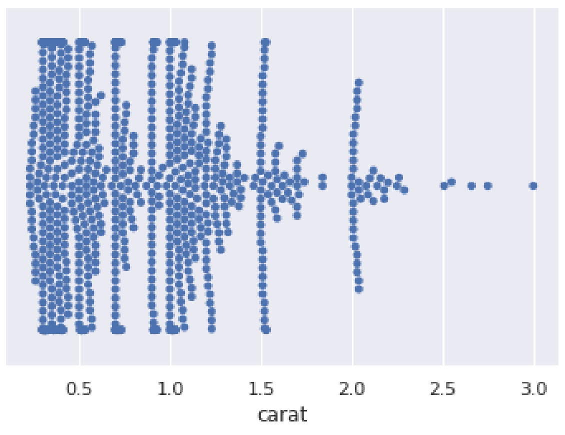


Swarm plot

The points plotted represent individual data points.

1. Swarm plot for single variable using diamonds dataset

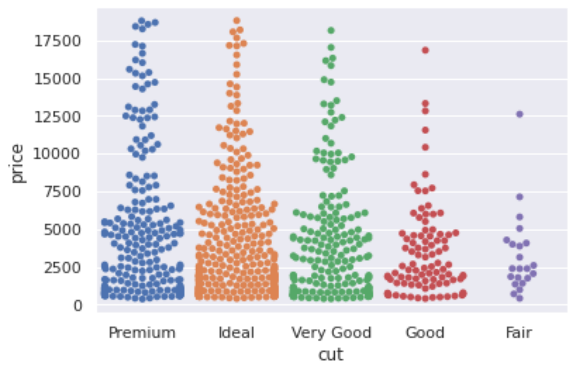
```
# a multimodal distribution is observed
sns.swarmplot(d.sample(1000).carat);
```



2. Swarm plots for two variable using diamonds dataset

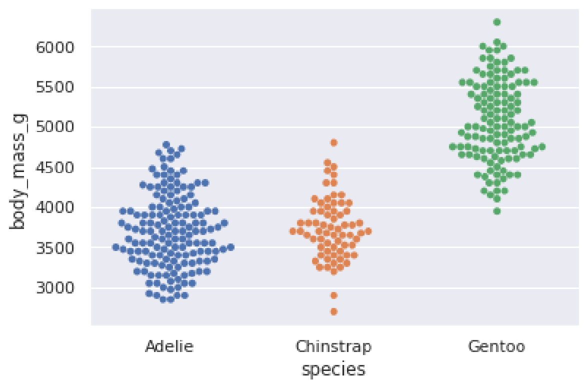
- The plot is categorical vs continuous variable. The distribution of a categorical variable over a continuous variable can be observed.

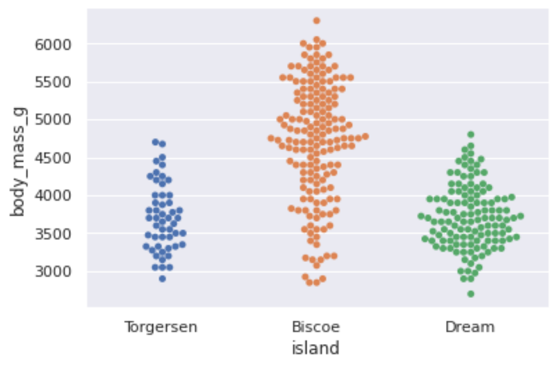
```
# first plot
sns.swarmplot(x='cut', y='price', data=d.sample(1000));
```

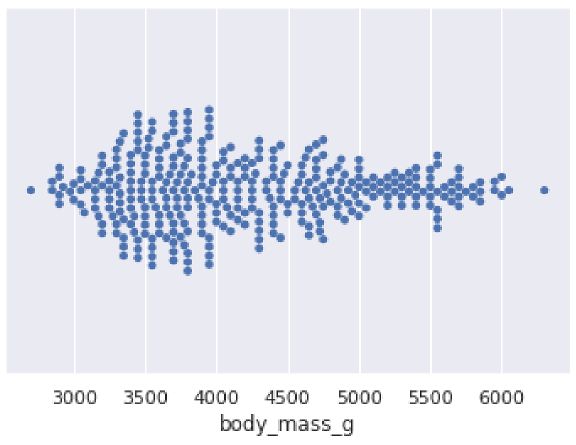


3. Example plots using penguin dataset

```
# loading penguins dataset
p = sns.load_dataset('penguins')
# first plot
sns.swarmplot(x='species', y='body_mass_g', data=p);
# second plot
sns.swarmplot(x='island', y='body_mass_g', data=p);
# third plot
sns.swarmplot(x='body_mass_g', data=p);
```







- From the plots it can be observed that the Gentoo species has larger body mass, thus the Biscoe Island must contain the Gentoo species along with one of the other two.
- The third plot shows the skewness in the distribution of body mass towards the left.

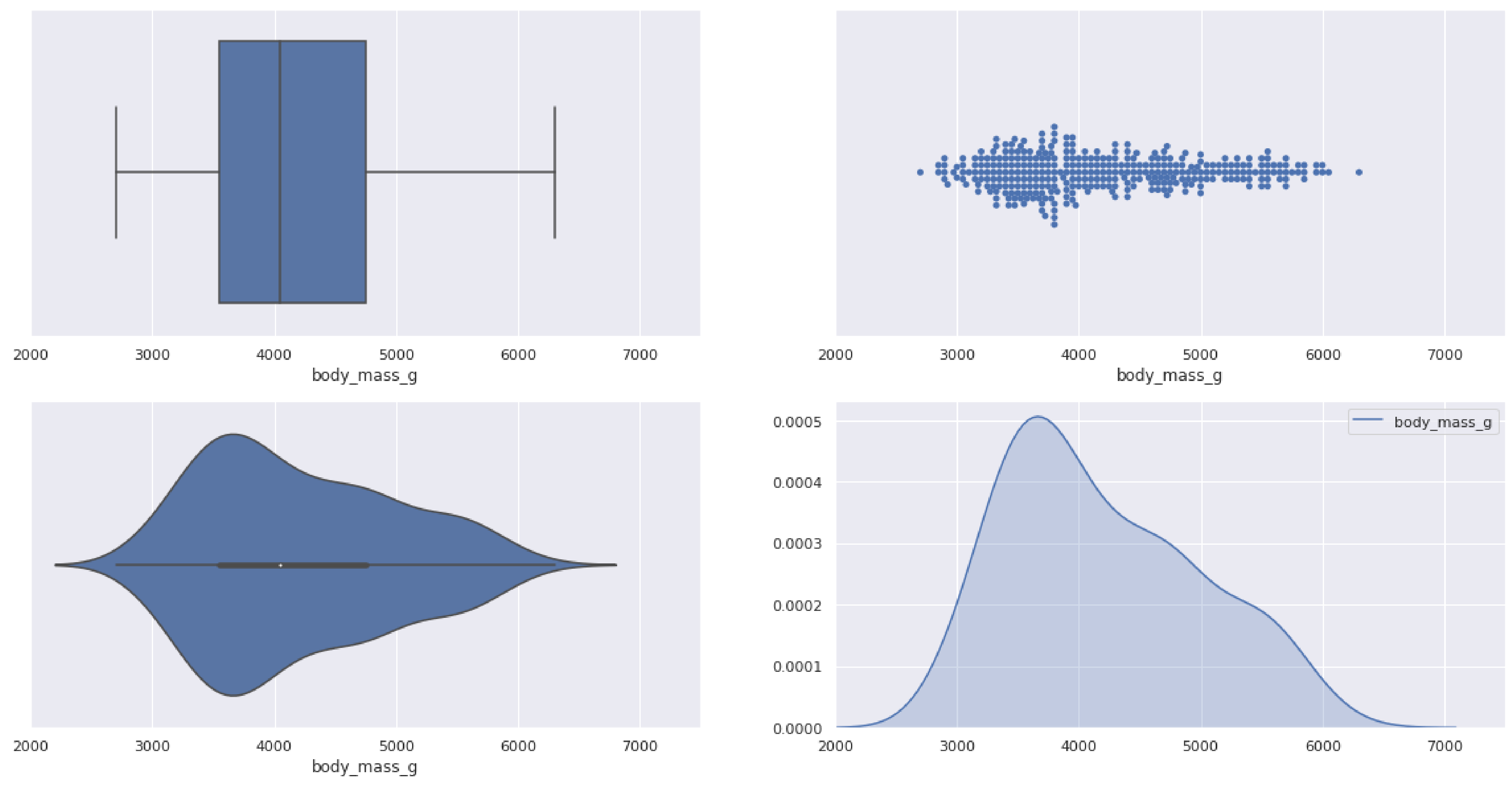
Violin plot

- The median, inter quartile region and whisker like line are marked out along with a kde plot showing the data distribution.

- The drawbacks are that the individual points (like in swarm plot) and the outliers (as in box plot) cannot be visualised in a violin plot.

1. Comparison: violin plot vs box plot vs kde plot vs swarm plot (for a single variable).

```
# creating subplots
fig, axs = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(20, 10);# setting the size
p1 = sns.swarmplot(x='body_mass_g', data=p, ax=axs[0][1],);
p1.set(xlim=(2000, 7500));# setting the x axis limits
p2 = sns.violinplot(x='body_mass_g', data=p, ax=axs[1][0]);
p2.set(xlim=(2000, 7500));
p3 = sns.boxplot(x='body_mass_g', data=p, ax=axs[0][0]);
p3.set(xlim=(2000, 7500));
p4 = sns.kdeplot(p.body_mass_g, shade=True, ax=axs[1][1]);
p4.set(xlim=(2000, 7500));
```



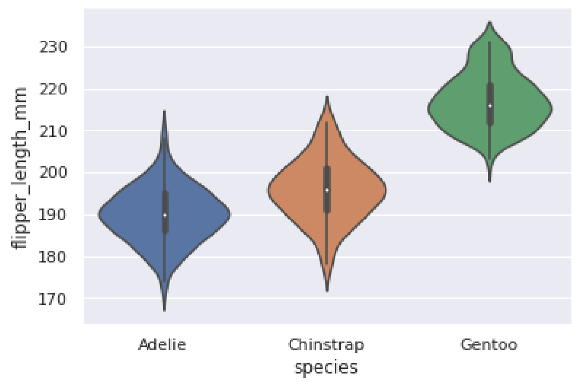
It can be seen that the violin plot combines the visual information from the other plots.

Multiple violin plots

- To plot multiple violin plots, a vertical orientation of plots is used.
- Multiple modes in a distribution might be a hint to multiple distributions, which can be further drilled down.

1. Plotting discrete variable vs continuous variable.

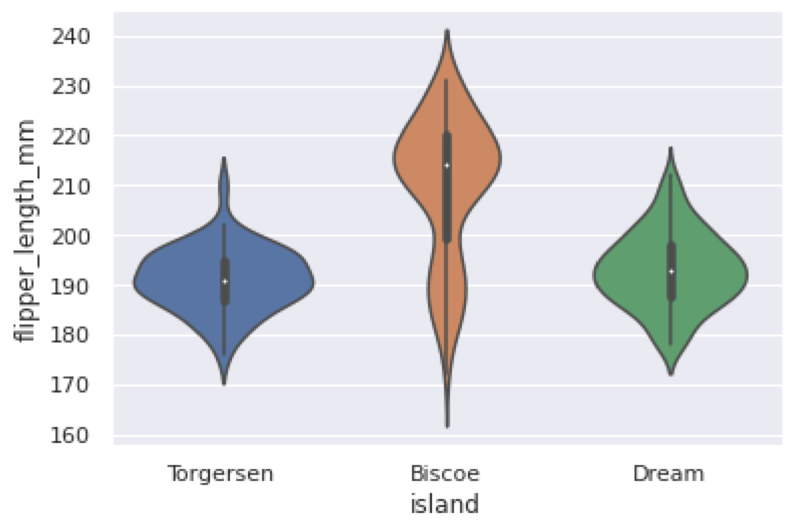
```
sns.violinplot(x='species', y='flipper_length_mm', data=p);
```

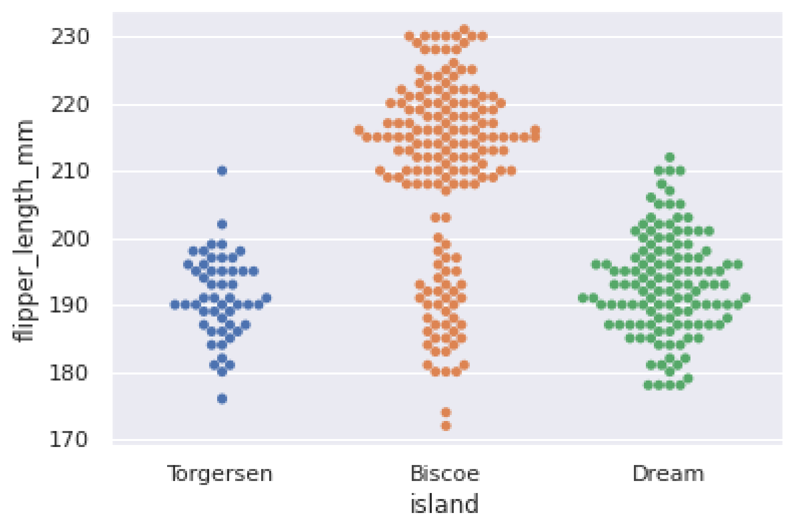


2. Comparison with swarm plot

- The multimodal distribution in violin plots can be visualised as a cluster of points in swarm plots.

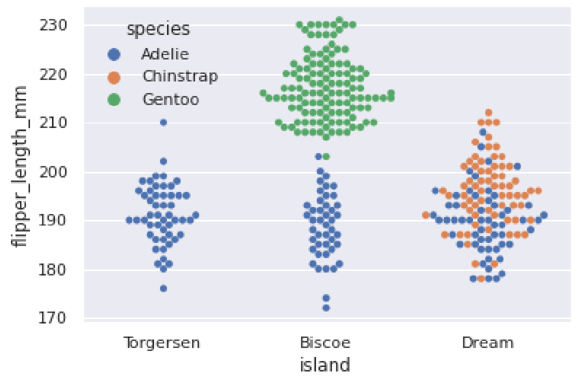
```
# first plot
sns.violinplot(x='island', y='flipper_length_mm', data=p);
# second plot
sns.swarmplot(x='island', y='flipper_length_mm', data=p);
```





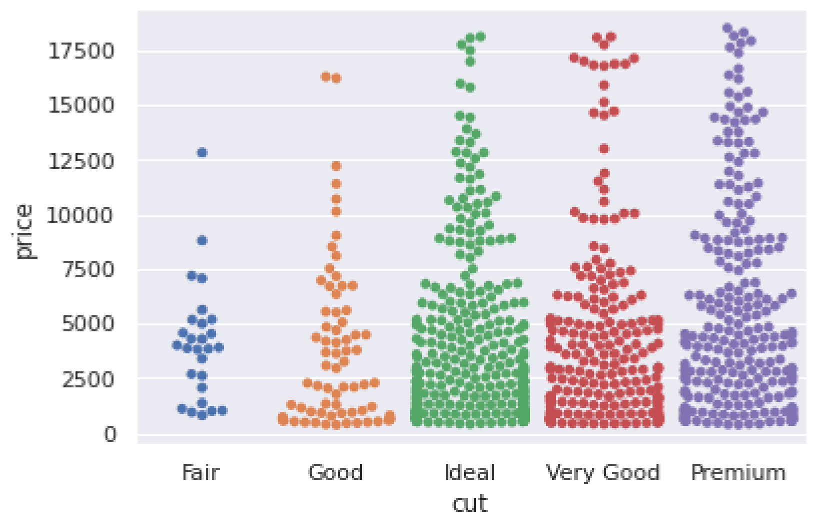
- Another categorical dimension called hue can be added to the swarmplots. Thus, some more details can be embedded in the same plot.

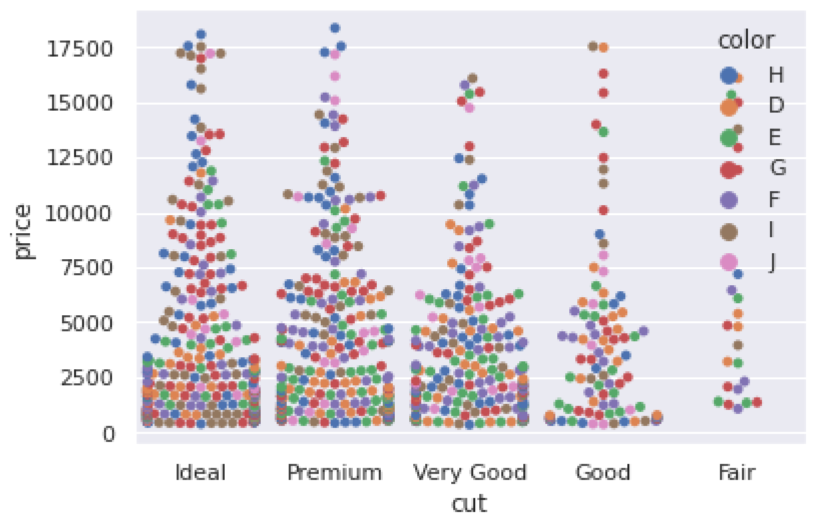
```
sns.swarmplot(x='island', y='flipper_length_mm', hue='species', data=p);
```



- To visualize the cause of variation in a small dataset, swarms plots are better than kde plots. If there are two many categories, the distribution in kde plot is better for visualization.

```
# first plot
sns.swarmplot(x='cut', y='price', data=d.sample(1000));
# second plot
sns.swarmplot(x='cut', y='price', hue='color', data=d.sample(1000));
```

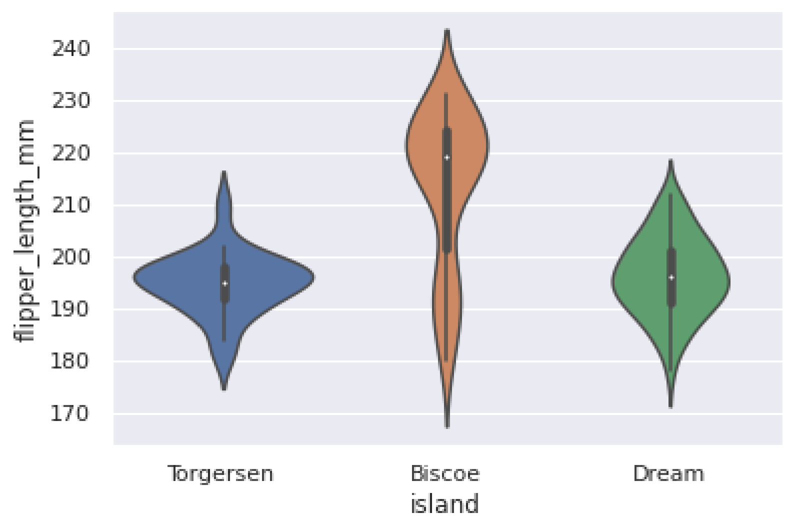


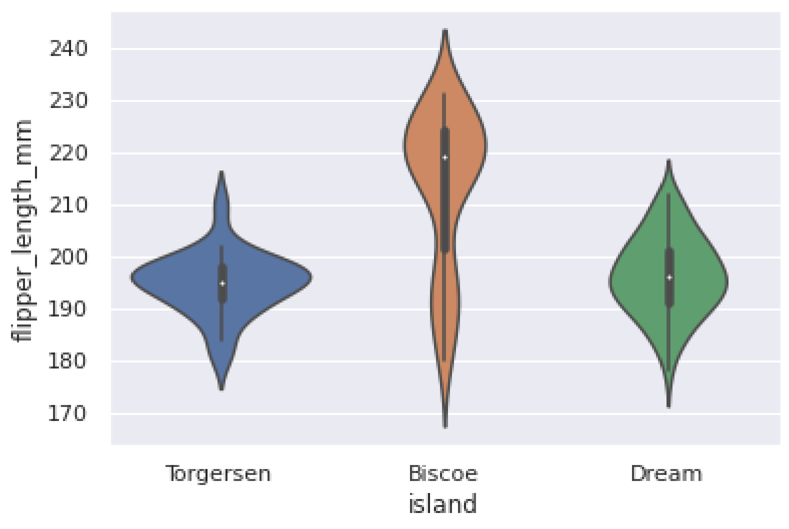


Paired violin plot

1. Adding additional categorical variables

```
# first plot - male
sns.violinplot(x='island', y='flipper_length_mm', data=p[p.sex=='MALE']);
#second plot - female
sns.violinplot(x='island', y='flipper_length_mm', data=p[p.sex=='FEMALE']);
```

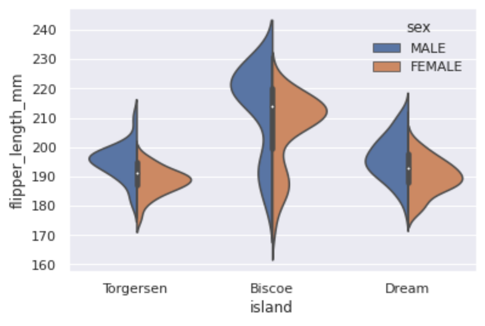




2. Paired violin plots - with split

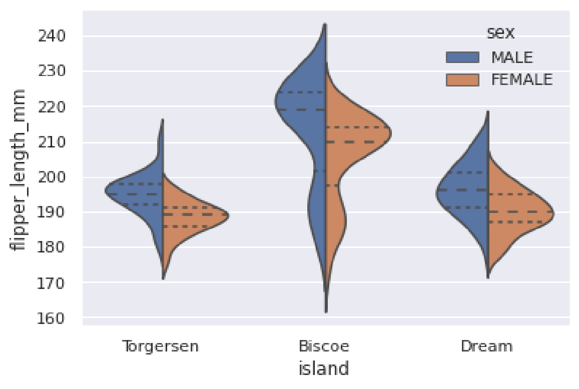
- The kde is plotted for the two categories but the median and IQ region is marked without considering this split.
- There must be exactly two levels to split a violin plot.

```
sns.violinplot(x='island', y='flipper_length_mm', hue='sex', split=True, data=p);
```



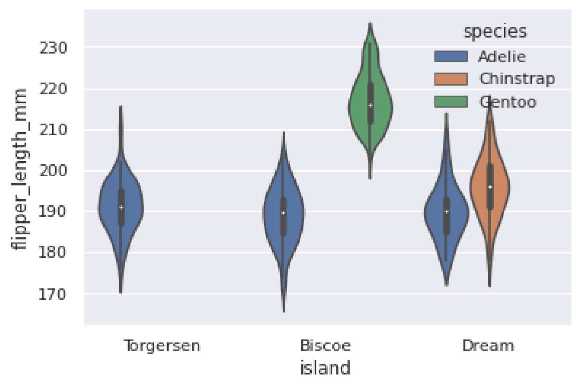
3. Paired plots with seperate marking of median median and inter quartile region.

```
sns.violinplot(x='island', y='flipper_length_mm',
               hue='sex', split=True, inner='quartile', data=p);
```



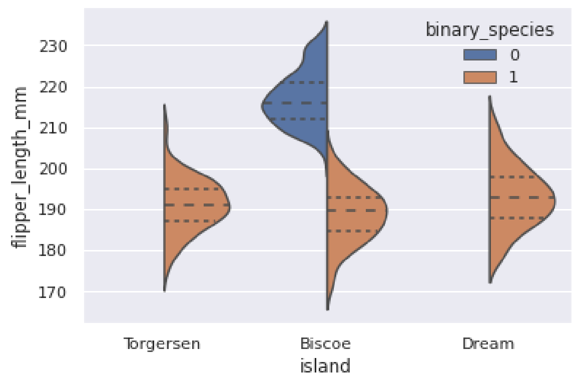
4. Paired violin plots - without split

```
sns.violinplot(x='island', y='flipper_length_mm',
               hue='species', data=p);
```



sex5. Converting multi categorical to binary variable.

```
p['binary_species'] = p.species.apply(lambda x: 0 if x == 'Gentoo' else 1)
sns.violinplot(x='island', y='flipper_length_mm',
               hue='binary_species', split=True, inner='quartile', data=p);
```

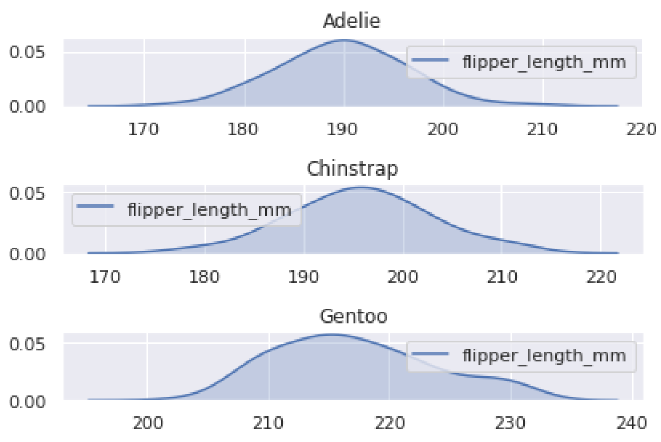


Faceted plot

Faceted plots are used for combining sets of plots using one or more categories.

1. Plotting multiple plots.

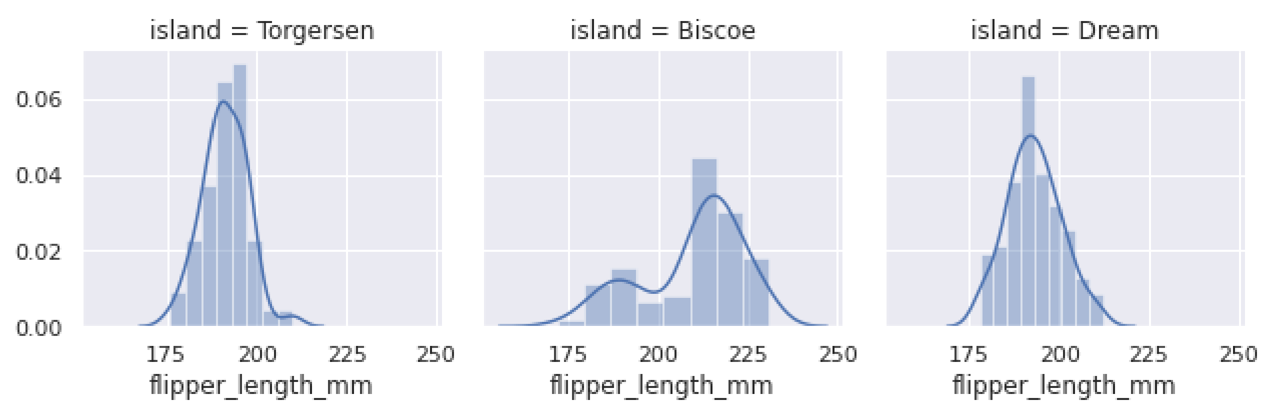
```
# plotting multiple kde plots
column_name = 'species'
nrows = len(p[column_name].unique())
fig, axs = plt.subplots(nrows=nrows);
i = 0
for i,c_v in p[column_name].unique():
    pl = sns.kdeplot(p[p[column_name] == c_v].flipper_length_mm,
                shade=True, ax=axs[i]);
    pl.set_title(c_v); # to set the title of each plot
    i += 1
plt.tight_layout() # to get overlapped plot diagrams
```



2. Using faceted grids

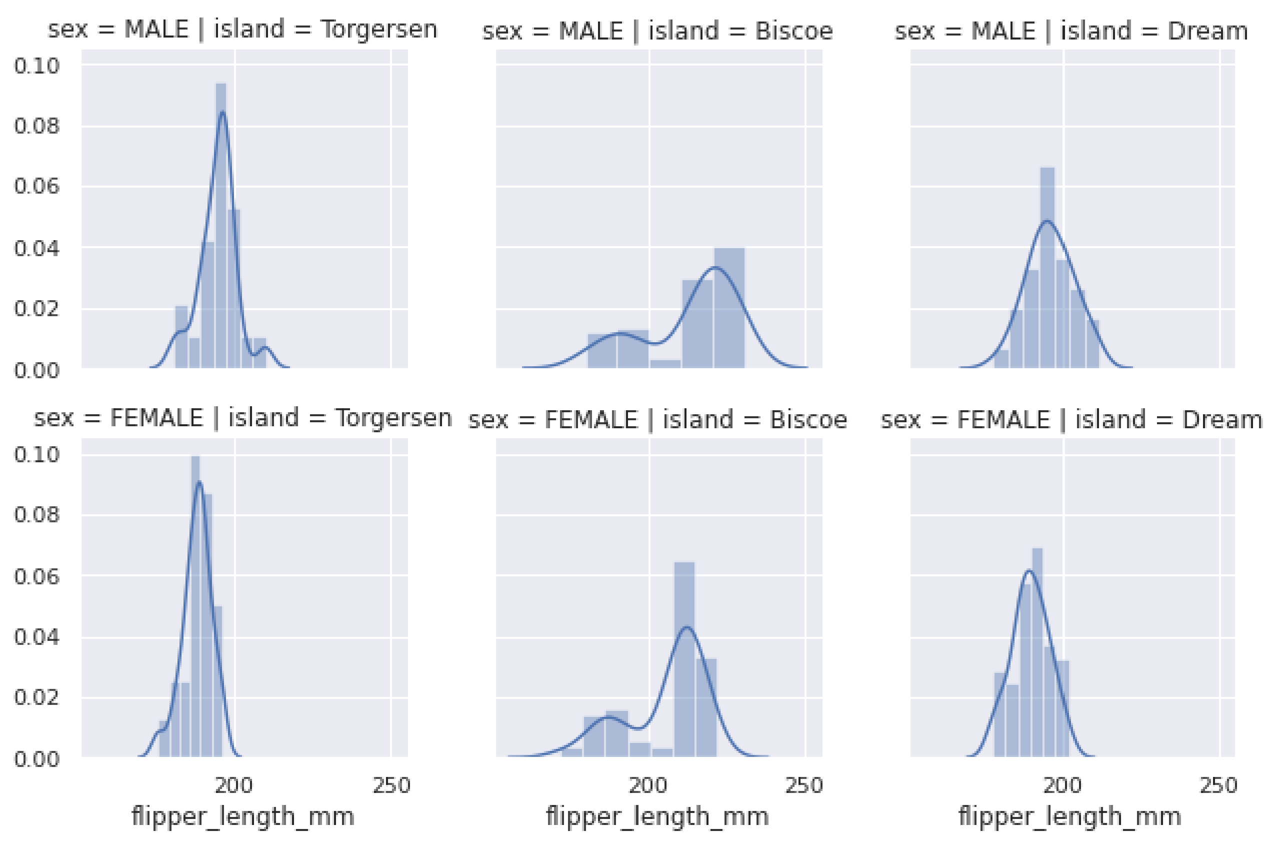
- The axis is aligned automatically along with addition of title. The axis is marked only on one side depending on the grid orientation.

```
g = sns.FacetGrid(p, col='island');
g.map(sns.distplot, 'flipper_length_mm');
```



- The grids can be split on rows/columns as done above or both(shown below):

```
g = sns.FacetGrid(p, col='island', row='sex');
g.map(sns.distplot, 'flipper_length_mm');
```

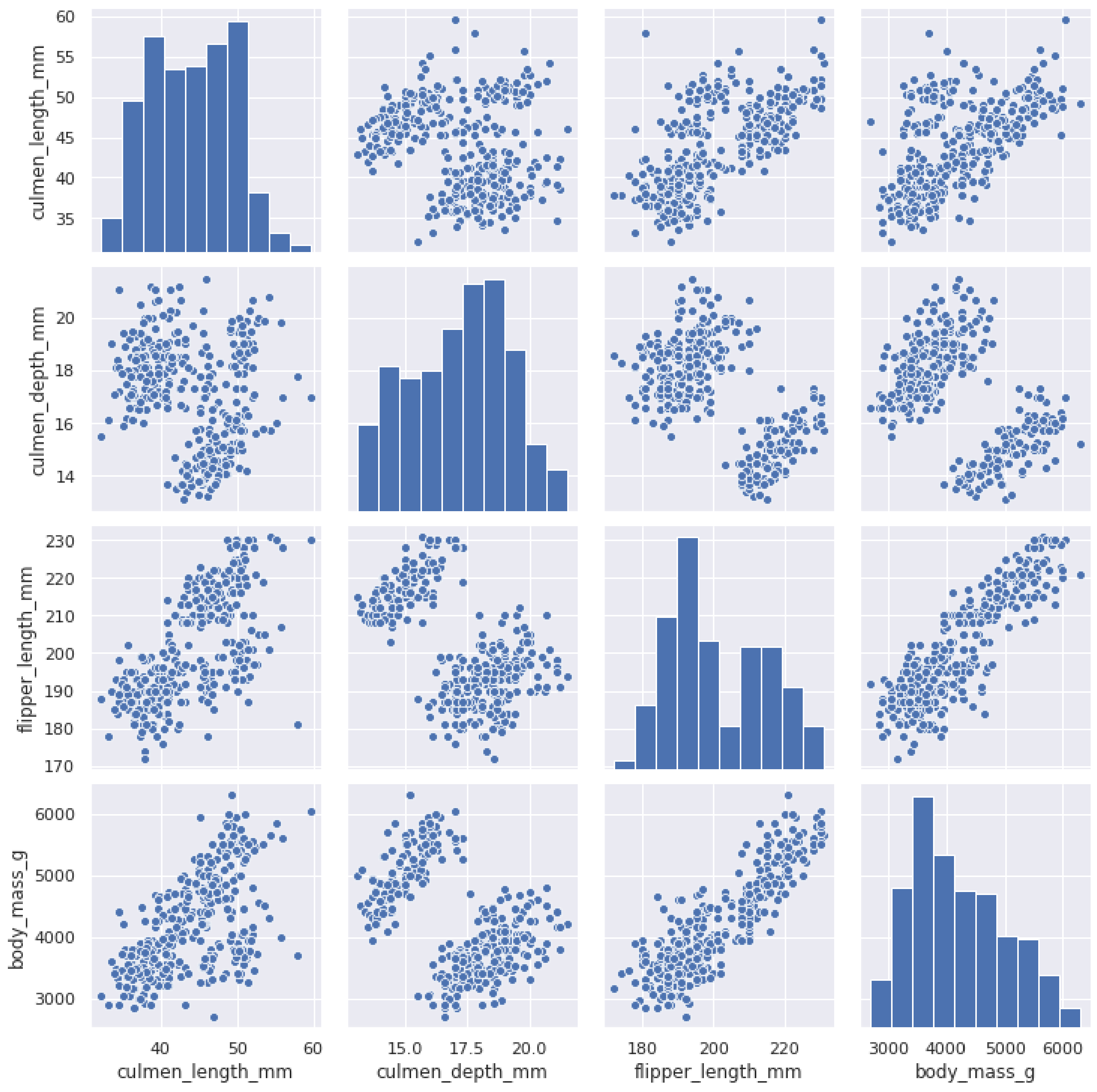


Pair plot

1. Scatterplots - without hue

- Pair plots are used for visualisation of relationship between quantitative variables, exhaustively.i.e for all attribute pairs.
- The histogram of each variable is plotted along with the scatter plot of all variable pairs.
- Thus, the data distribution of each variable (the skewness, modes etc) can be realised using the histograms whereas, the variable relationships and data clusters can be seen in the scatter plots.

```
# eg using penguin dataset
sns.pairplot(p);
```



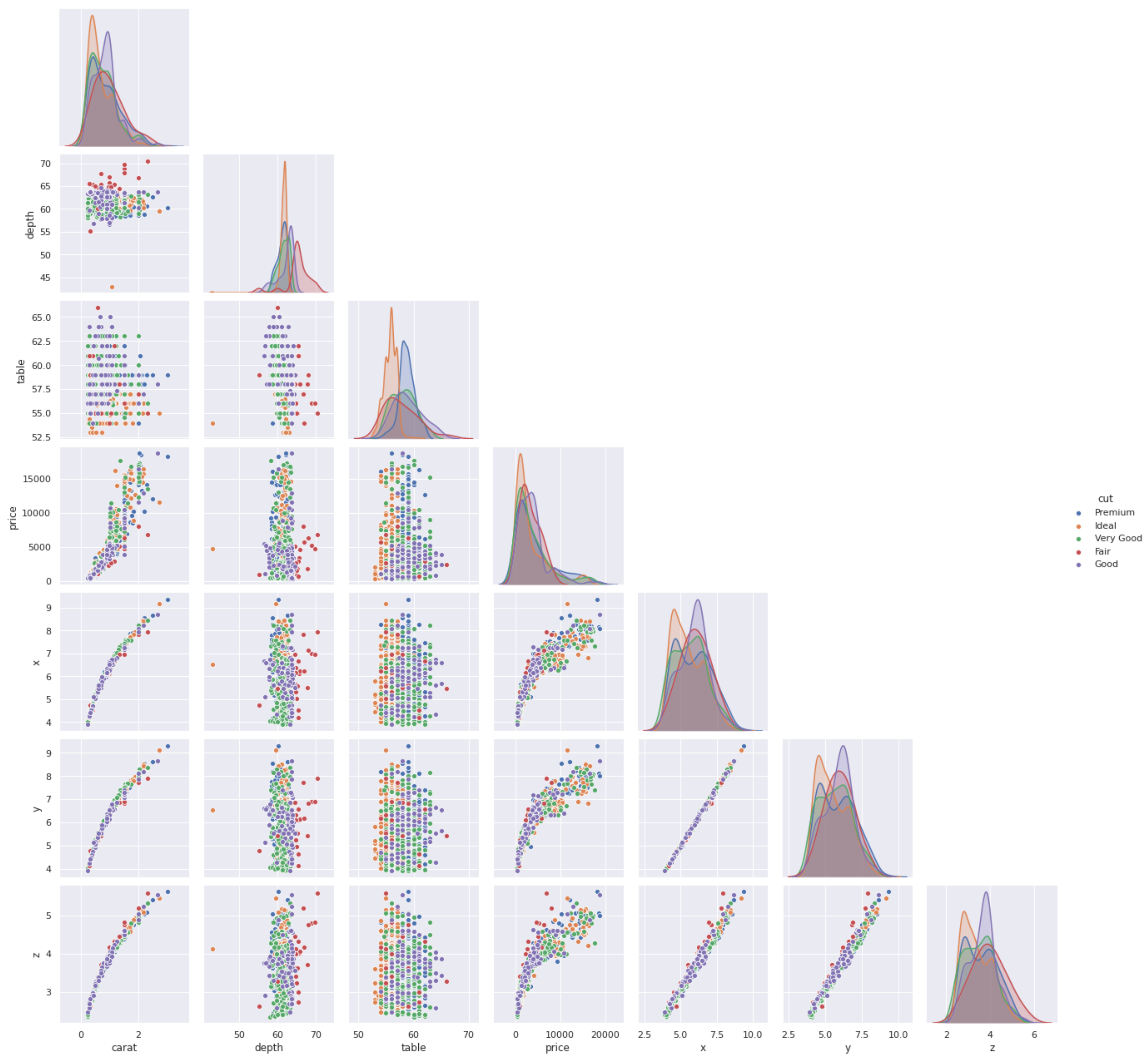
2. Scatter Plots - with hue

- The distribution shown is using a kde plot.

3. Example using subset of diamonds data

- The relationship between variables can be linear - e.g. x vs y , quadratic - e.g x vs price or the variables may not be correlated at all - e.g. table vs price.
- The replication of plots can be avoided by setting the corner variable to True. the distribution of values in each plot along a given categorical variable can be plotted using the parameter hue.

```
sns.pairplot(d.sample(1000), hue='cut', corner=True);
```



Summary

- Tables can be styled to visualise the data. For example the cells can be coloured, gradient shading of cells or bars of values relative to a given cell axis can be drawn.
- The distribution of a single continuous variable can be visualised using histograms (bars, rugs and kde), box plots and boxen lots. Boxen plots help visualise the distribution beyond the inter quartile range of the box plots.
- Bar plots are used for categorical variables
- The joint distribution of two variables can be seen using distplots and pair plots, using scatter plots and distplots.
- Swarm plots and violin plots can be used to visualise the distribution of a single continuous variable or continuous vs discrete variables.
- The violin plots have the median marked, IQR and the kde plot showing the data distribution.
- Additional dimensions can be added to the swarm plot by specifying the hue and the violin plots using hue and split.
- Faceted plots can be used to visualise multiple plots across a given axis.